



Module Federation Quick Start Guide

What is Module Federation?

Module Federation is a technology that makes it easier to share code between JavaScript applications and makes team collaboration more efficient.

It can be used to build distributed architectures across both frontend and backend stacks and enables Micro Frontends, Team Autonomy, Independent Deployments and Faster Local Development Performance.

On the back of this card are Quickstart Guides for getting started with Module Federation with Nx for React and Angular apps.

Check out our docs for more...

For a deeper dive into Module Federation with Nx, take a look at our docs. Relevant topics, including micro-frontend architecture and build optimization, are collected at:

nx.dev/concepts/module-federation

Connect with us!

-  Website: <https://nx.dev>
-  Github: <https://github.com/nrwl/nx>
-  Discord: <https://go.nrwl.io/community>
-  YouTube: <https://www.youtube.com/@nxdevtools>
-  X: <https://twitter.com/nxdevtools>



React Quickstart Guide

Step 1

Create a new Nx Workspace for your Module Federation architecture

```
~/
npx -y create-nx-workspace@latest acme --preset=apps
```

Step 2

Add the `@nx/react` plugin to your Nx workspace

```
~/acme/
npx nx add @nx/react
```

Step 3

Scaffold your host and remote applications with the `@nx/react:host` generator

```
~/acme/
npx nx g @nx/react:host shell --remotes=remote1,remote2
```

Step 4

Serve your Module Federation setup with a single command:

```
~/acme/
npx nx serve shell
```

This will serve the host application with HMR and the remotes statically (no file watching).

Step 5

When actively developing on a specific remote specify `--devRemotes` to setup HMR for that remote

```
~/acme/
npx nx serve shell --devRemotes=remote1
```



Angular Quickstart Guide

Step 1

Create a new Nx Workspace for your Module Federation architecture

```
~/
npx -y create-nx-workspace@latest acme --preset=apps
```

Step 2

Add the `@nx/angular` plugin to your Nx workspace

```
~/acme/
npx nx add @nx/angular
```

Step 3

Scaffold your host and remote applications with the `@nx/angular:host` generator

```
~/acme/
npx nx g @nx/angular:host shell --remotes=remote1,remote2
```

Step 4

Serve your Module Federation setup with a single command:

```
~/acme/
npx nx serve shell
```

This will serve the host application with HMR and the remotes statically (no file watching).

Step 5

When actively developing on a specific remote specify `--devRemotes` to setup HMR for that remote

```
~/acme/
npx nx serve shell --devRemotes=remote1
```